

JC544 U.S. PTO

UTILITY PATENT APPLICATION TRANSMITTAL

Assistant Commissioner for Patents
Washington, D.C. 20231
BOX PATENT APPLICATIONAttorney Docket No.: PA990485
Date: December 21, 1999
Express Mail Label No.: EJ380218563US

Dear Sir:

Transmitted herewith for filing is the patent application of:

Inventor(s): Rose, et al.For: METHOD OF AUTHENTICATING ANONYMOUS USERS WHILE REDUCING POTENTIAL FOR "MIDDLEMAN" FRAUD

Enclosed are:

1. ☒ Patent application (9) total pages.
2. ☒ Drawings: ☐ Formal () sheet(s) or ☒ Informal (1) sheet(s).
3. ☒ Declaration/Power of Attorney: ☐ Signed ☒ Unsigned
4. ☐ An Assignment () pages and Recordation Form Cover Sheet.
5. ☐ A Preliminary Amendment () pages.
6. ☐ Information Disclosure Statement (IDS):
 - a. ☐ PTO-1449
 - b. ☐ Copies of IDS Citations (number of citations:)
7. ☐ Other:

CLAIMS:	(a) Filed	(b) Extra Claims	Large Entity Fee	Fee Paid
Total*	20 - 20	0	x \$18 =	\$0.00
Independent**	4 - 3	1	x \$78 =	\$78.00
Multiple Dependent Claim(s): <input type="checkbox"/> No <input type="checkbox"/> Yes			\$260	\$
APPLICATION FILING FEE			\$760	\$760.00
			TOTAL FEE	\$838.00

*If the number in column a is less than 20, enter 0 in column b.

**If the number in column b is less than 3, enter 0 in column b

8. ☐ A check in the amount of \$_____ is enclosed to pay the filing fee.
9. ☒ Please charge Deposit Account No. 17-0026 of QUALCOMM Incorporated the amount of \$838.00. The Commissioner is hereby authorized to charge payment of any additional fees which may be required, or credit any overpayment, to said Deposit Account No. 17-0026. A duplicate of this sheet is enclosed for fee processing.
10. ☒ The Commissioner is further hereby authorized to charge to said Deposit Account No. 17-0026, pursuant to 37 CFR 1.25(b), any fee whatsoever which may become properly due or payable, as set forth in 37 CFR 1.16 to 37 CFR 1.18 inclusive, for the entire pendency of this application without specific additional authorization.

Date: December 21, 1999Signature: Thomas R. Rouse

Thomas R. Rouse, Reg. No. 40,793

QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, California 92121-1714
Telephone: (858) 651-6732
Facsimile: (858) 658-2502JC503 U.S. PTO
09/468557
12/21/99

METHOD OF AUTHENTICATING ANONYMOUS USERS WHILE REDUCING POTENTIAL FOR "MIDDLEMAN" FRAUD

BACKGROUND OF THE INVENTION

5

I. Field of the Invention

The present invention pertains generally to the field of encryption, and more particularly to methods for authenticating anonymous users that reduce the potential for "middleman" fraud.

10

II. Background

Computer software is generally distributed over the Internet to end users (i.e., consumers) by distribution agents, or "middlemen." There are thus three parties involved in the transaction. The first is the provider of software and content (i.e., the author), who derives revenue from providing content to end users and pays a small commission to distribution agents to promote and distribute the software. The second is one of a number of distribution agents who provides the software (which provides a mechanism to view the content, as well as some value independent of the content, such as, e.g., electronic mail functions) to the user. The middleman derives revenue from users who receive content, so it is in his or her interest to distribute the software widely. The third is the user, who gets the software for free in return for viewing the content. The user gets no other remuneration, mainly because the users are anonymous. The users are anonymous because tracking details are generally not kept and the users have not been individually identified. Users might volunteer information when requesting content, but such volunteered information is generally used and discarded rather than being stored or tracked. The parties are hereinafter referred to generally as the provider, the middleman, and the user.

25
30

The middleman may use a device known as an Internet "cookie" to obtain demographic information about users when the users connect to the Internet and visit the appropriate website. For example, when a user connects to certain Internet locations, the user's computer connects through the Internet to a host computer operated by the middleman. The host sends a small data file (the cookie) that is saved by the user's computer. As the user and the host communicate, some

35

data is stored in the cookie. When the user disconnects, the cookie remains in his or her computer. Subsequent data about the user's Internet use is stored in the cookie. The next time the user connects to the host, the host reads the cookie for information about the user. The user's information may be compiled by the host operator and sold to Internet marketers.

Because the users are anonymous, the middleman can commit undetectable fraud on the provider simply by passing through more content. This, in turn, can be accomplished either by requesting more content on behalf of a real user, or by creating "fake" users. There are a number of well-known statistical methods for tracking the rate of the content delivery to particular users while keeping the details anonymous. Such statistical methods solve the problem of middlemen committing fraud by requesting more content on behalf of a real user. However, these methods are not directed to the situation in which the middleman commits fraud by creating a significant number of fake users. Thus, there is a need for a method of preventing software distribution agents from impersonating a significant number of non-existent users to commit fraud.

SUMMARY OF THE INVENTION

The present invention is directed to a method of preventing software distribution agents from impersonating a significant number of non-existent users to commit fraud. Accordingly, in one aspect of the invention, a method for a provider of software to authenticate users of the software is provided. The method advantageously includes the steps of constructing a puzzle in response to information received from a user, the puzzle including the information; sending the puzzle to the user; and returning a solution to the puzzle to the provider.

In another aspect of the invention, an apparatus for enabling a provider of software to authenticate users of the software is provided. The apparatus advantageously includes means for constructing a puzzle in response to information received from a user, the puzzle including the information; means for sending the puzzle to the user; and means for returning a solution to the puzzle to the provider.

In another aspect of the invention, an apparatus for enabling a provider of software to authenticate users of the software is provided. The apparatus advantageously includes a processor; and a processor-readable storage medium accessible by the processor and containing a set of instructions executable by the

processor to construct a puzzle in response to information received from a user, the puzzle including the information, and send the puzzle to the user.

In another aspect of the invention, a method of preventing a person from impersonating a plurality of users of software is provided. The method advantageously includes the steps of constructing a plurality of puzzles, each puzzle having a solution that includes information about a respective one of the plurality of users, each puzzle requiring consumption of a resource to solve; and sending each puzzle to a respective one of the plurality of users for solution.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a system in which information is exchanged between a provider and a user.

FIG. 2 is a block diagram illustrating the fit of bits of an encrypted "cookie" into an exponentiation operation of a "puzzle."

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the embodiments described below, new users of a system are registered in a way that uses up a scarce resource, so that an individual user will register and not notice the cost, but any party, and in particular a middleman, would incur significant costs in any attempt to misuse the system. In one embodiment the significant resource is computation time. Other scarce resources could be used, such as, e.g., storage space, network bandwidth, or user attention span (i.e., requiring a user to interact for a period of time). Computation time is preferred because it is essentially free to the user. Generally the user's personal computer is idle most of the time, the computation can be performed in a non-intrusive manner, and there is no ongoing overhead once the computation is complete. The exact amount of computation can be adjusted according to parameters such as the cost and computing power of typical computers at the time, and the amounts involved in potential fraud.

A system 100 in which a user 102 requests registration by communicating to a provider 104 is illustrated in FIG. 1 in accordance with one embodiment. It should be understood that the user 102 and the provider 104 refer to machines used by the user and the provider, such as, e.g., personal computers or handheld computing devices such as personal digital assistants or wireless telephones for the

user 102, up to large Web servers for the provider 104. In a message 106 the user 102 sends demographic data to the provider 104. In other embodiments the user 102 may send other data such as, e.g., user identity instead of or along with the demographic data in the message 106. The message 106 to the provider 104 may or may not be in an encrypted format to protect sensitive information about the user 102. The software on the user 102 includes some identifier of the middleman, which, in turn, is also included in the message 106. The user 102 may also provide information to better target content material subsequently. The data 106 provided by the user 102 will be encoded by the provider 104 into the answer to a puzzle 108. The answer, i.e., the decrypted puzzle 110, will be communicated back to the provider 104 by the user 102 at a later time when the user 102 requests content.

In one embodiment the answer 110 to the puzzle 108 is constructed as follows. The information 106 provided, and a random identifier (not shown) of sufficient length to assume uniqueness, are placed in a buffer (also not shown). A cryptographically secure hash function of the contents of the buffer is computed. An exemplary secure hash function is the Secure Hash Standard specified in Federal Information Processing Standard (FIPS) 180-1, produced by the National Institute for Standards and Technology. The algorithm described in the FIPS 180-1 document is hereafter referred to as the Secure Hash Algorithm (SHA). The result of the hash function may be 160 bits in length. In a particular embodiment, only the first sixty-four bits of the hash function are used. In another embodiment the first eighty bits of the hash function are used. The hash function is inserted into the buffer at the beginning of the buffer. The entire buffer is then encrypted with a symmetric block cipher such as, e.g., the Data Encryption Standard (DES) algorithm (FIPS 46-2), using a key-held secret (not shown) in the provider 104. In a particular embodiment, the triple-DES (3DES) algorithm as specified in the draft text of FIPS 46-3 is used.

To everyone except the provider 104, the result of the encryption is advantageously indistinguishable from random data. In particular, even if some of the data is made public, no party can reconstruct the part that is not exposed. When the provider 104 subsequently receives the encrypted item, the provider 104 can be assured that it is exactly the data that the provider 104 created at an earlier time by decrypting it and verifying that the hash function embedded in it agrees with the result of a new computation. The encrypted buffer of data is commonly referred to as a "cookie" by those skilled in the art.

It should be understood that the cookie will be longer than a single cipher block, so the encryption should be done in Cipher Block Chaining mode, as described in FIPS. Cipher Block Chaining mode normally requires a random Initialization Vector. However, the initialization vector can be set to a constant zero value because the first block to be encrypted contains a hash value, which is sufficient to thwart expected attacks.

The puzzle 108 is constructed from the cookie instead of transmitting the cookie itself. The puzzle 108 is advantageously constructed such that a certain (i.e., expected) amount of computation is required to solve the puzzle 108 and recover the cookie. Because the provider 104 has to perform this function for many users, the provider 104 must be computationally efficient to construct the puzzle 108 while also being intentionally inefficient to solve the puzzle 108. This combined computational construction efficiency and solution inefficiency can be accomplished with a novel use of known public-key cryptographic methods.

For purposes of the following discussion, the cookie may be denoted C , and additional parameters P and g are embedded in the software and hence known to all participants. The parameter P is advantageously a large prime number. The lower bound on the number of bits in P is constrained by the desired computational complexity of the puzzle 108. In a particular embodiment the number of bits in P is 1024. The parameter P advantageously has the additional property that a parameter Q , which satisfies the equation $Q = (P-1)/2$, is also prime. The parameter g is advantageously a generator of the subgroup of order Q of the multiplicative group of integers modulo P . The parameters P , g , and Q are typical parameters used for the well-known Diffie-Hellman key agreement protocol. If the cookie, C , is larger than Q , the user 102 will be given some of the cookie plainly, and a smaller part will be used to construct the puzzle 108. It is generally assumed herein that $|C| < |Q|$.

The puzzle 108 is first constructed by computing $Z = g^k \text{ modulo } P$. Currently, the best known method for computing K given Z (i.e., for solving the Discrete Logarithm Problem) is computationally expensive, and with a 1024-bit length for P , is considered to be roughly equivalent to decrypting a message using a block cipher with an unknown key of 128 bits. Such a computation is too extensive for the user 102 to perform. Because the user 102 is supposed to be able to recover C , the provider 104 gives the user 102 most of the answer 110. The puzzle 108 thus includes Z and a puzzle hint. The puzzle hint includes most (but not all) of the information about C . The number of bits of variability in the value transmitted

determines the difficulty of the puzzle 108. The most efficient way for the user 102 to solve the puzzle 108 is to try out guesses for the unknown information until the user 102 finds the guess that yields the correct answer 110. Checking each guess requires computing the modular exponentiation function for the candidate K .

It can be assumed that computing such a modular exponentiation takes about $1/100^{\text{th}}$ of a second (the actual time depends on the speed of the processor (not shown) and the size of P). If it is desired that the computation take an average of twelve hours of background processing time, approximately four million (or 2^{22}) trial candidates must be used. Because, on average, the solution will be found about half way through the set of possibilities, the puzzle hint should consist of all but twenty-three of the bits of the answer 110.

To ensure that the puzzle 108 cannot be solved in some manner avoiding trial exponentiation, a one-way hash function may again be used. Suppose (as is the case for the Secure Hash Standard) that the output of the hash function $H()$ is 160 bits in length. It is important to ensure that $|C|$ is somewhat larger than $|H|/2$ so $|C|$ can be split into two parts. If necessary, C can be padded before encryption to ensure that $|C|$ is large enough. The result of the hash function is used in part to obscure C and in part to vary the input to the exponentiation operation.

An intermediate result K is constructed from C in the following manner. The cookie C is divided into two parts, L and R , such that $|R|$ is eighty bits in length. A random number r is chosen in the range $0, \dots, N$, where N determines the average difficulty of the puzzle 108. In one embodiment $N = 2^{23}$. Then K is determined by the following equation: $K = L || ((R || 0_{80}) \oplus H(L || r))$, where $||$ denotes a concatenation operation, \oplus denotes a bitwise Exclusive-Or (XOR) operation, and 0_{80} denotes eighty zero-bits.

It should be noted that in the particular embodiment being described, the result of a single hash function is advantageously split into two parts and used for independent purposes. It would be readily apparent to one of skill in the art that two independent hash functions could be used for these purposes.

The puzzle 108 then consists of the exponentiation result Z and all but the last eighty bits of K . To recover C and solve the puzzle 108, the user 102 starts trying values of r , calculating $H(L || r)$, appending eighty bits of the result to the given part of K , and checking whether the resulting g^K is equal to the answer Z . When the correct r is found, the left eighty bits of the hash output may be XORed with the partial K to recover C .

Hence, the above-described technique satisfies the requirements as stated. The possible solutions of the discrete logarithm problem vary in 160 bits, far too many for any form of precomputation to be useful. Eighty bits of C are obscured until K can be verified by trial exponentiation. Eighty bits of K are not revealed until they are derived from r . Because K depends on L , there is no way to precompute the limited set of useful hash values. The range of the random number r determines the average time to solve the puzzle 108 by trial and error, while the above properties ensure that other shortcuts do not work.

It would be readily apparent to one of skill in the art that the puzzle 108 could also be solved by sending tries to the provider 104 and waiting for an acknowledgment. The accompanying protocol should ensure that this is not more efficient than performing the modular exponentiation (which, in practice, will be satisfied).

Once the user 102 has solved the puzzle 108, the user 102 is in possession of a valid cookie that contains enough information to subsequently convince the provider 104 that the user 102 has registered. The cookie also carries any ancillary data required by the provider 104 to determine the content.

It should be understood that some of the information in the initial registration request 106 is potentially privacy-sensitive. Similarly, when the cookie is returned to the provider 104 for subsequent content requests, an eavesdropper could track the requests based on the cookie. Therefore, it is desirable that the communication 106 from the user 102 to the provider 104 be encrypted, and it is relatively easy to accomplish such an encryption using a discrete-logarithm-based, public-key encryption algorithm such as, e.g., the Diffie-Hellman algorithm. The public key of the provider 104 could be embedded in the application, and the common P and g parameters could be used. Nevertheless, it would be understood by those of skill in the art that the message 106 from the user 102 to the provider 104 need not be encrypted, and that in the event the communication 106 is encrypted, any public-key encryption algorithm could be used.

In one embodiment an encrypted cookie is created and then decrypted, and a puzzle is created from the encrypted cookie and then solved, as described below and with reference to FIG. 2. In accordance with this particular embodiment, the external environment is as follows. Certain data and functionality are assumed to be present in the calling environment. In particular, more functionality is required from the provider than from the user.

The primary common parameters of the puzzle system are P , which is 1024 bits in length and prime, and g , a generator whose value is two. The prime, P , is given by $2^{1024} - 2^{960} - 1 + 2^{64} * \{ [2^{694}\pi] + 129093 \}$. The hexadecimal value of P is the following:

5

```

FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08
8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B
302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9
A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6
49286651 ECE65381 FFFFFFFF FFFFFFFF.

```

10

The following common functionality is advantageously used. Both the user and the provider require access to an SHA-1 hash algorithm and a long integer modular exponentiation function. The SHA-1 hash algorithm, which is denoted $H(.)$, advantageously produces 160-bit results from any input, regardless of the length of the input. The long integer modular exponentiation function implicitly references g and P above, and efficiently calculates g^x modulo P for a given value x . Other long integer modular arithmetic functions, such as, e.g., addition, multiplication, etc., may be used instead in the implementation of the modular exponentiation functionality.

20

The following provider-side-only parameters and functionality are advantageously employed. In addition to the above-described common functionality requirements, the provider additionally requires the ability to create and verify cookies in a cryptographically secure manner. To do this, the provider requires the following items: (1) the 3DES encryption algorithm; (2) an authentication key Ka ; (3) a key Kp used to encrypt cookies; and (4) a source of high-quality secret pseudo-random numbers. In one embodiment the authentication key Ka is 128 bits in length. It should be understood that encryption algorithms other than the 3DES encryption algorithm may be employed. In the embodiment in which the 3DES encryption algorithm is used, the key Kp for encrypting cookies should advantageously be 112 bits in length.

25

30

In accordance with this particular embodiment, an encrypted cookie may be created as follows. In the specific embodiment described, the cookie is a byte buffer. The maximum length of the input cookie is assumed to be limited by the 1024-bit length of the prime P plus eighty bits that will eventually be appended. The cookie is padded with one or more octets to be a multiple of eight octets, and

35

has eight octets of authentication information prepended. With the eighty additional bits that are appended, the length of the cookie must remain less than 1023 bits. Therefore, in this particular embodiment, the input cookie must be at most 101 octets, or 808 bits, in length. It would be understood by those of skill in the art that longer cookies can be handled with slight modifications if required. The cookie should advantageously have a minimum length of eight octets. The following pseudo-code parameters may be applied to encrypt a cookie in accordance with this specific embodiment:

```

10 void encryptCookie(
    const unsigned char *cookie,
    int cookieLength,
    unsigned char *encryptedCookie
);

```

15 In the above pseudo-code parameters, the parameter *cookie* points to a buffer with *cookieLength* (at most 101) octets of information. The parameter *encryptedCookie* points to a buffer with at least $((\text{cookieLength} + 16) \& 0xF8)$ octets of space available, which will be overwritten with the result. The global information used includes the parameters *Ka* and *Kp*.

20 The following method steps may be used to encrypt a cookie in accordance with this specific embodiment. First, the SHA hash $H(Ka, \text{cookie})$ is calculated and then truncated to eight octets. The truncated hash is then copied into a file called *encryptedCookie*. Second, a file called *cookie* is appended into the *encryptedCookie* file.

25 Third, the number of octets of padding required, $1 \leq n \leq 8$, is calculated, and then the number of octets that contain the value *n* are appended. This third method step can be unambiguously reversed. Fourth, the *encryptedCookie* file is encrypted using the 3DES encryption algorithm with the key *Kp* in Cipher Block Chaining (CBC) mode and an Initialization Vector of zero.

30 In accordance with this particular embodiment, an encrypted cookie may be decrypted by using the following pseudo-code parameters:

```

int decryptCookie(
    const unsigned char *encryptedCookie,
35 int cookieLength,
    unsigned char *cookie
)

```

);

In the above pseudo-code parameters, the file *encryptedCookie* points to a buffer with *cookieLength* (at most 112) octets of information. The file *cookie* points to a buffer with at least (*cookieLength* - 8) octets of space available, which will be overwritten with the result. The return value is the length of the decrypted cookie if the authentication succeeds. Otherwise, the return value is zero. The global information used includes the parameters *Ka* and *Kp*.

The following method steps may be used to decrypt an encrypted cookie in accordance with this specific embodiment. First, a file called *encryptedCookie* is copied into a temporary buffer and then decrypted using the 3DES encryption algorithm with the key *Kp* in CBC mode and an Initialisation Vector of zero. Second, the SHA hash $H(Ka, \text{buffer}+8)$ is calculated after removing padding. The SHA hash is then truncated to eight octets. Third, the eight-octet output is compared with the first eight octets of the buffer. The value zero is returned if the two compared sets of eight octets are unequal. Fourth, if the comparison is equal, the temporary buffer plus eight octets is copied to the *cookie* file, and the unpadding of the *cookie* file is returned.

In accordance with this particular embodiment, a puzzle may be created from an encrypted cookie as illustrated in FIG. 2. An encrypted cookie 200 is conceptually partitioned into left (*L*) and right (*R*) components. The right component is eighty bits in length. A random number generator 202 pseudo-randomly selects a number in the range from zero to 2^{25} . The encrypted cookie 200 is copied into a 1024-bit buffer 204 and padded on the left with zeros 206 and padded on the right with ten octets of zeros (eighty zero bits). A 160-bit, SHA hash function 208 is performed on the concatenation of the left component of the encrypted cookie 200 and the random number selected by the random number generator 202. The result of the hash function 208, which is twenty octets in length, is bitwise XORed into the rightmost twenty octets of the buffer 204, modifying the rightmost ten octets of the encrypted cookie 200 and ten octets of zeros. The buffer 204 is exponentiated to produce an exponentiated result *Z*. The exponentiated result *Z* is sent to the user, or client, in the puzzle 210, which includes the result *Z* and all but the rightmost eighty bits of the buffer 204, which are discarded.

In accordance with one embodiment, the following pseudo-code structure may be used to create a puzzle from an encrypted cookie:

```

struct puzzle {
    int            difficulty;
    int            cookieLength;
    unsigned char  answer[128];
5    unsigned char encryptedCookie[cookieLength];
}

```

```

void makePuzzle(
    unsigned char *encryptedCookie,
10    int cookieLength,
    int difficulty,
    struct puzzle *p
);

```

- 15 The input *difficulty* determines the size of the random number used, which in turn influences the expected number of trial exponentiations that need to be done to solve the puzzle. If *difficulty* is, for example, twenty, then on average 2^{19} hash calculations and trial exponentiations will need to be done by the user to break the puzzle. The input encrypted cookie has length *cookieLength*. This function returns after filling in the structure pointed to by *p*. The *encryptedCookie* in the structure is different from the input encrypted cookie, and will therefore fail to authenticate.
- 20

The encrypted cookie is conceptually split into two parts *L* and *R*, so that *R* is ten octets (eighty bits) long. A random number *r* is generated in the range $0 \dots 2^{\text{difficulty}} - 1$. The encrypted cookie is then copied into a 1024-bit temporary buffer *K*, which is padded on the left with zeros, and padded with ten octets of zeros at the right end. A hash function of *L*, *r* is performed to produce a value *h*, which is twenty octets long. The value *h* is XORed into the last twenty octets of the buffer, modifying the last ten octets of the original encrypted cookie and the other ten octets of zeros. The temporary buffer *K* is treated as a 1024-bit integer and

30 exponentiated to produce *Z* according to the following equation: $Z = g^Z \bmod P$. The pseudo-code structure is filled in by pointing to the appropriate fields according to the following pseudo-code steps:

```

35    p->difficulty = difficulty;
    p->cookieLength = cookieLength;
    p->answer = Z;

```

p->encryptedCookie = the middle part of K.

In accordance with one embodiment, a puzzle may be solved by calling a software routine denoted *solvePuzzle*. The puzzle solution function must be called repeatedly to actually solve the puzzle. Each call performs one trial exponentiation. The calling program is advantageously given the responsibility for functions such as, e.g., creating background threads, saving the intermediate state periodically, etc., as would be understood by those skilled in the art. The following pseudo-code structure completely defines the state of the search process, and is the information that needs to be saved and restored to continue:

```
struct puzzlestate {
    struct puzzle p;
    int upto;
    unsigned char intermediate[128];
};
```

```
int solvePuzzle(struct puzzlestate *s);
```

The *solvePuzzle* routine should return a value of one when the *solvePuzzle* routine has found the solution to the puzzle, in which case the encryptedCookie field of the puzzlestate structure will contain a valid encrypted cookie. While still searching, the *solvePuzzle* routine should return a value of zero. In the "impossible" case that the *solvePuzzle* routine has not found the valid encrypted cookie before searching the entire range, the *solvePuzzle* routine should return a value of negative one. Such a result could only occur in the case in which the transmission of the puzzle gets corrupted or there is a bug in the program at either the user end or the provider end.

It should be pointed out that before calling *solvePuzzle* for the first time, the user should copy the puzzle received from the provider into the above-shown puzzlestate structure. Also before calling *solvePuzzle* for the first time, the user should set the field *upto* to zero.

It should also be noted that it would be understood by those of skill in the art that various alternative methods could be used to solve the puzzle and verify the correctness of the cookie. One such method is the use of keyed message authentication codes.

It is important that the method used to solve the puzzle be efficient. Even though the goal is to use computer time, it is important that the time used should be unavoidable, and not subject to simple optimization. For this reason, the first call to *solvePuzzle* must calculate an intermediate result and save the result to avoid subsequent computation (which is certainly what someone who wanted to break the system would do). Because $g^{(a \cdot b)} = (g^a)^b$, it is possible to break the puzzle guess up into fixed and variable parts, and compute the exponentiation only on the smaller, variable part. In fact, by dividing (multiplying by the inverse) the answer by the fixed part, it is necessary only to exponentiate the 160-bit variable part and then compare to check whether the problem is solved.

In accordance with this particular embodiment, the puzzle may be solved by performing the following steps: First, if the *upto* field is zero, the *intermediate* field is initialized. The initialization procedure is performed by splitting the *encryptedCookie* field into left and right parts (*L* and *R*) such that *R* is ten octets long, and then calculating the multiplicative inverse of ($L \cdot 2^{160}$) and multiplying by the *answer* field to get the resultant value for the *intermediate* field. Second, a hash function is performed on the *L* and *upto* fields, and a 160-bit result is formed from *R* and the rightmost ten octets of the hash. Third, the 160-bit result is exponentiated to produce an exponentiation result. Fourth, the exponentiation result is compared to the value in the *intermediate* field. If the compared values are different, the *upto* field is incremented and a value of zero is returned. (Or, if the value of the *upto* field is greater than or equal to $2^{\text{difficulty}}$ (i.e., something has gone wrong), a value of negative one is returned.) Fifth, otherwise (i.e., if the compared values in the fourth step are the same), the leftmost eighty bits of the hash of *L* and *upto* are XORed into the rightmost bits of the *encryptedCookie* field (which is now correct), and a value of one is returned, indicating success.

Thus, a novel and improved method of authenticating anonymous users while reducing potential for "middleman" fraud has been described. Those of skill in the art would understand that the various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. The various illustrative components, blocks, modules, circuits, and steps have been described generally in terms of their functionality. Whether the functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans recognize the interchangeability of hardware and software under

these circumstances, and how best to implement the described functionality for each particular application. As examples, the various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented or performed with a digital

5 signal processor (DSP), an application specific integrated circuit (ASIC), discrete gate or transistor logic, discrete hardware components such as, e.g., registers and FIFO, a processor executing a set of firmware instructions, any conventional programmable software module and a processor, or any combination thereof. The processor may advantageously be a microprocessor, but in the alternative, the

10 processor may be any conventional processor, controller, microcontroller, or state machine. The software module could reside in RAM memory, flash memory, ROM memory, registers, hard disk, a removable disk, a CD-ROM, or any other form of storage medium known in the art. Those of skill would further appreciate that the data, instructions, commands, information, signals, bits, symbols, and

15 chips that may be referenced throughout the above description are advantageously represented by voltages, currents, electromagnetic waves, magnetic fields or particles, optical fields or particles, or any combination thereof.

Preferred embodiments of the present invention have thus been shown and described. It would be apparent to one of ordinary skill in the art, however, that

20 numerous alterations may be made to the embodiments herein disclosed without departing from the spirit or scope of the invention. Therefore, the present invention is not to be limited except in accordance with the following claims.

CLAIMS

What is claimed is:

1. A method for a provider of software to authenticate users of the
2 software, comprising the steps of:
3 constructing a puzzle in response to information received from a
4 user, the puzzle including the information;
5 sending the puzzle to the user; and
6 returning a solution to the puzzle to the provider.
- 7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205

- 10 result to generate an exclusive-OR result, and concatenating the first component
and the exclusive-OR result to produce the value.

6. The method of claim 4, wherein the exponentiating step comprises
2 the steps of raising a generator to a power, the power being the derived value,
dividing the generator raised to the power of the derived value by a prime
4 number, and obtaining the remainder of the division operation.

7. An apparatus for enabling a provider of software to authenticate
2 users of the software, comprising:
means for constructing a puzzle in response to information received
4 from a user, the puzzle including the information;
means for sending the puzzle to the user; and
6 means for returning a solution to the puzzle to the provider.

8. The apparatus of claim 7, wherein the information comprises
2 demographic information about the user.

9. The apparatus of claim 7, wherein the information comprises an
2 identity of the user.

10. The apparatus of claim 7, wherein the means for constructing a
2 puzzle comprises means for deriving a value from the information to produce a
derived value, means for exponentiating the derived value to produce an
4 exponentiated value, and means for combining the exponentiated value with a
portion of the derived value.

11. The apparatus of claim 10, further comprising means for storing the
2 information and a random number, means for performing a hash function on the
information and the random number to generate a first hash result, and means for
4 encrypting the first hash result, wherein the means for deriving means for
partitioning the encrypted hash result into first and second components,

- 6 performing a hash function on a concatenation of the first component and the
random number to generate a second hash result, appending a plurality of zero
8 values to the second component to produce a lengthened second component,
performing an exclusive-OR operation between the lengthened second component
10 and the second hash result to generate an exclusive-OR result, and concatenating
the first component and the exclusive-OR result to produce the value.

12. The apparatus of claim 10, wherein the means for exponentiating
2 comprises means for raising a generator to a power, the power being the derived
value, means for dividing the generator raised to the power of the derived value
4 by a prime number, and means for obtaining the remainder of the division
operation.

13. An apparatus for enabling a provider of software to authenticate
2 users of the software, comprising:
a processor; and
4 a processor-readable storage medium accessible by the processor and
containing a set of instructions executable by the processor to construct a puzzle in
6 response to information received from a user, the puzzle including the information,
and send the puzzle to the user.

14. The apparatus of claim 13, wherein the information comprises
2 demographic information about the user.

15. The apparatus of claim 13, wherein the information comprises an
2 identity of the user.

16. The apparatus of claim 13, wherein the puzzle is constructed by
2 deriving a value from the information to produce a derived value, exponentiating
the derived value to produce an exponentiated value, and combining the
4 exponentiated value with a portion of the derived value.

17. The apparatus of claim 16, wherein the set of instructions is further
2 executable by the processor to store the information and a random number,
4 perform a hash function on the information and the random number to generate a
6 first hash result, and encrypt the first hash result, wherein the derived value is
8 derived by partitioning the encrypted hash result into first and second
10 components, performing a hash function on a concatenation of the first component
and the random number to generate a second hash result, appending a plurality of
zero values to the second component to produce a lengthened second component,
performing an exclusive-OR operation between the lengthened second component
and the second hash result to generate an exclusive-OR result, and concatenating
the first component and the exclusive-OR result to produce the value.

18. The apparatus of claim 16, wherein the exponentiated value is
2 exponentiated by raising a generator to a power, the power being the derived
4 value, dividing the generator raised to the power of the derived value by a prime
number, and obtaining the remainder of the division operation.

19. A method of preventing a person from impersonating a plurality of
2 users of software, comprising the steps of:
constructing a plurality of puzzles, each puzzle having a solution that
4 includes information about a respective one of the plurality of users, each puzzle
requiring consumption of a resource to solve; and
6 sending each puzzle to a respective one of the plurality of users for
solution.

20. The method of claim 19, wherein the resource is computer processing
2 time.

ABSTRACT

5 A method of authenticating anonymous users while reducing potential for "middleman" fraud includes the step of constructing a puzzle in response to information received from a software user. The puzzle includes the received information. The puzzle is sent to the user by a software provider. The user solves the puzzle and returns the solution to the provider. The puzzle includes a portion of a value derived from an encrypted "cookie" and an exponentiation of the derived value. The cookie includes information about the user.

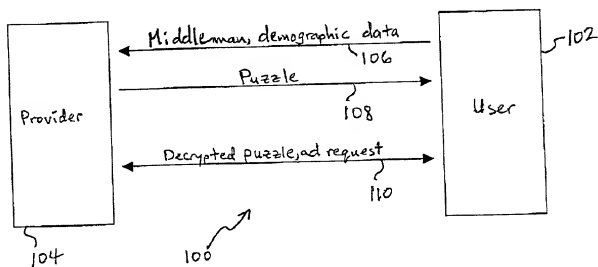


FIG. 1

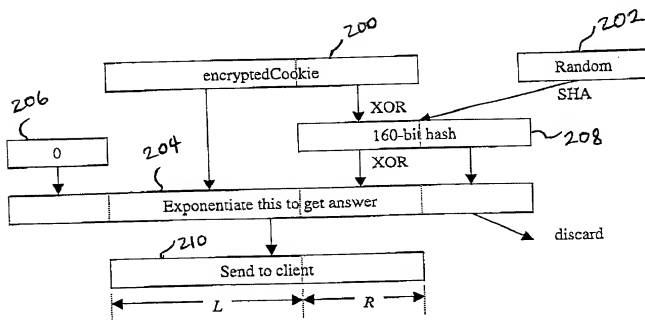


FIG. 2

COMBINED DECLARATION / POWER OF ATTORNEY

ATTORNEY DOCKET NO.: QCPA990485

AS BELOW NAMED INVENTOR, I HEREBY DECLARE THAT: This Declaration is of the following type:

☒ Original ☐ Supplemental ☐ Continuation-In-Part ☐ Divisional
Continuation National Stage of PCT

My residence, post office address and citizenship are as stated below next to my name: I believe I am the original, first and sole inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled **METHOD OF AUTHENTICATING ANONYMOUS USERS WHILE REDUCING POTENTIAL FOR "MIDDLEMAN" FRAUD** the specification of which:

☒ is attached hereto.
☐ was filed on _____ as Serial No. _____
☐ was amended on _____ (if applicable).
☐ was described and claimed in PCT International Application No. _____ filed on _____
and as amended under PCT Article 19
on _____

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Sec. 1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, Sec. 119 of any foreign application(s) for patent or inventor's certificate or of any PCT International application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT International application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed.

Priority Claimed

(Country)	(Application No.)	(Day/Month/Year/Filed)	(Yes)	(No)
-----------	-------------------	------------------------	-------	------

(Country)	(Application No.)	(Day/Month/Year/Filed)	(Yes)	(No)
-----------	-------------------	------------------------	-------	------

I hereby claim the benefit under Title 35 USC 120 of the United States application(s) listed below, and insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35 USC 112, I acknowledge the duty to disclose material information as defined in Title 37 CFR 1.56(a) which occurred between the filing date of the prior application and the national or PCT International filing date of this application:

(Serial No.)	(Filing Date)	(Status)
--------------	---------------	----------

(Serial No.)	(Filing Date)	(Status)
--------------	---------------	----------

I hereby appoint the following attorneys and/or agents to prosecute this application and to transact all business in the U.S. Patent and Trademark Office connected therewith: Russell B. Miller, Reg. No. 31,122, Gregory D. Ogrod, Reg. No. 30,880, Brian S. Edmonston Reg. No. 38,864, Charles D. Brown, Reg. No. 28,285, Thomas R. Rouse, Reg. No. 40,793, Bruce W. Greenhaus, Reg. No. 37,339, Kent Baker Reg. No. 38,822, Tom Streeter Reg. No. 32,007, Thomas M. Thibault, Reg. No. 42,281, Kyong H. Macek, Reg. No. 42,977, Byron Yafuso, Reg. No. 45,244, Roger W. Martin, Reg. No. 39,291, Pavel Kalousek, Reg. No. 44,178, Philip R. Wadsworth, Reg. No. 29,219, Michael D. Hartogs, Reg. No. 36,547 and/or Sean English, Reg. 37,319. Please direct all telephone calls to Russell B. Miller at (858) 658-4833 and address all correspondence to: Russell B. Miller, QUALCOMM Incorporated, 5775 Morehouse Drive, San Diego, California 92121-1714.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

COMBINED DECLARATION / POWER OF ATTORNEY

ATTORNEY DOCKET NO.: QCPA990485

Full Name of Sole or First Inventor GREGORY G. ROSE	Inventor Signature	Date
Residence 6 Kingston Avenue, Mortlake, NSW 2137, AUSTRALIA	Citizenship Australia	
Post Office Address 6 Kingston Avenue, Mortlake, NSW 2137, AUSTRALIA		
Full Name of Second Inventor PHILIP HAWKES	Inventor Signature	Date
Residence 2/6-8 Belmore Street, Burwood, NSW 2134, AUSTRALIA	Citizenship Australia	
Post Office Address 2/6-8 Belmore Street, Burwood, NSW 2134, AUSTRALIA		

QUALCOMM Incorporated
 5775 Morehouse Drive
 San Diego, California 92121-1714
 Telephone: (858) 658-4833
 Facsimile: (858) 658-2502